

Programovací jazyk IMPERATOR

Dokumentace

[illegible]

Bc. Marek Čačka

Bc. Petra Talandová

Brno, 2005

Obsah

1	Jazyk iMPerator	5
1.1	Filosofie jazyka iMPerator	5
1.1.1	Vznik jazyka	5
1.1.2	Prvky jazyka	5
1.2	Abeceda jazyka iMPerator	7
2	Navržení lexikálního analyzátoru	8
2.1	Navržení jednotlivých dílčích gramatik	8
2.1.1	Nevýznamové znaky	8
2.1.2	Významové znaky	9
2.2	Sestavení lexikálního analyzátoru, spojení dílčích gramatik . .	11
2.2.1	Nevýznamové jednotky (L_n)	11
2.2.2	Významové jednotky (L_v)	11
2.2.3	Sloučení významových a nevýznamových jednotek . . .	11
2.3	NKA	12
2.4	DKA	14
3	Syntaktická analýza	15
3.1	Syntaktické diagramy	15
3.2	Gramatika pro syntaktickou analýzu	16
3.3	Výpočet množin First a Follow, ověření LL(1)	18
3.4	Rozkladová tabulka	21
4	Jak programovat v jazyku iMPerator	26
5	Implementace	27
6	Závěr	29
7	Příloha – testovací soubory	30
7.1	Kalkulačka	30
7.2	Pyramida	31
7.3	Násobilka	32
7.4	Faktoriál cyklem	32
7.5	Faktoriál rekurzí	33
7.6	Hanojské věže	34

1 Jazyk iMPerator

1.1 Filosofie jazyka iMPerator

1.1.1 Vznik jazyka

Našemu jazyku jsme dali jméno iMPerator. To naznačuje nejen skutečnost, že jde o imperativní jazyk, ale vyjadřuje i obdiv ke starověké kultuře a k historii. Z tohoto důvodu jsme také zavedli latinské pojmenování klíčových slov, protože latina je, stejně jako angličtina, jazyk mezinárodně srozumitelný.

1.1.2 Prvky jazyka

Jazyk iMPerator bude podporovat následující prvky programovacích jazyků: čísla celá a desetinná, znaky a řetězce (tedy skalární proměnné), identifikátory proměnných, aritmetické a logické výrazy, přiřazovací příkaz, rozhodovací příkaz, programový cyklus, uživatelské funkce a vstupně-výstupní funkce. Dále zde budou „podpůrné prostředky“, jako jsou oddělovač příkazů, programový blok, tzv. bílé znaky a dva druhy komentářů.

Jazyk iMPerator se částečně inspirovuje jazyky typu C / C++, ale přináší i vlastní originální konstrukce. Následující přehled ukazuje, jak jsou definovány jednotlivé prvky jazyka.

Komentáře. Řádkový komentář začíná znakem # a je ukončen koncem řádku. Víceřádkový komentář je uzavřen do hranatých závorek.

```
# komentář ↵  
[ komentář ]
```

Bílé znaky. Do bílých znaků patří mezera a tabulátor.

Identifikátory proměnných. Proměnné začínají i končí otazníkem. Díky tomu je ve zdrojovém textu snadno rozpoznatelná a navíc intuitivně vyjadřuje proměnlivost obsahu proměnné. Proměnná může obsahovat nejen čísla a písmena, ale i další znaky včetně mezery, přičemž musí mít aspoň jeden znak. Vyloučen je tabulátor a znak konce řádku.

```
?ident.1?
```

Čísla. Jazyk obsahuje celá i desetinná čísla v desítkové soustavě, oddělovačem desetinné části je čárka. Jiné číselné soustavy nepodporujeme.

```
2,598
```


Cyklus. Podporován je cyklus s podmínkou na začátku. Po klíčovém slově DONEC následuje logická podmínka uzavřená v závorkách. Cyklus probíhá tak dlouho, dokud podmínka platí.

DONEC (podmínka) { blok }

Funkce. Funkce je uvozena klíčovým slovem FUNCTIO, po němž následuje název funkce a případně seznam parametrů. Za touto hlavičkou je blok příkazů, které se mají pro zdárné vykonání funkce provést.

FUNCTIO názevfunkce (par1; par2; ...) { blok }

Pro volání funkce se použije tato konstrukce:

názevfunkce (par1; par2; ...)

Vstupně-výstupní funkce. Jazyk obsahuje jednu funkci pro vstup (čtení) a jednu pro výstup (výpis). Funkce pro čtení je uvozena klíčovým slovem LECTO a v závorce je uveden název proměnné, do níž se načítá ze standardního vstupu. Pro výpis slouží funkce SCRIBO, u níž je za klíčovým slovem uvedeno, co je výstupem.

LECTO (proměnná)

SCRIBO (relační výraz)

Klíčová slova. Jsou psána vždy velkými písmeny, skládají se jen ze znaků A až Z. Názvy funkcí mohou obsahovat i malá písmena a čísla, musí však velkým písmenem začínat.

1.2 Abeceda jazyka iMPerator

Dílčí abecedy pro jednotlivé prvky jazyka uvádí následující seznam.

Komentáře:

$\Sigma_K = \{ \#, [,], \leftrightarrow, (\Sigma - \{ \leftrightarrow \}), (\Sigma - \{ [] \}) \}$

Výraz $(\Sigma - \{ \leftrightarrow \})$ je cokoli kromě znaku prázdného řádku, $(\Sigma - \{ [] \})$ je cokoli kromě pravé hranaté závorky, komentář tak může obsahovat cokoli. Tedy $\Sigma_K = \Sigma$.

Bílé znaky:

$\Sigma_M = \{ \sqcup, \text{TAB} \}$

Identifikátory:

$\Sigma_I = \{ ?, (\Sigma - \{ ?, \text{TAB}, \leftrightarrow \}) \}$

Výraz $(\Sigma - \{ ?, \text{TAB}, \leftrightarrow \})$ je cokoli kromě otazníku, tabelátoru a znaku konce řádku, tedy lze psát $\Sigma_I = \{ \Sigma - \{ \text{TAB}, \leftrightarrow \} \}$.

Čísla:

$\Sigma_N = \{+, -, , , 0..9\}$

Oddělovačem je desetinná čárka, povoleny jsou číslice 0 až 9.

Řetězce:

$\Sigma_S = \{ ", \backslash, (\Sigma - \{ ", \backslash\})\}$

Výraz $(\Sigma - \{ ", \backslash\})$ je cokoli kromě uvozovek a zpětného lomítka, tedy lze psát $\Sigma_S = \Sigma$.

Oddělovač příkazů:

$\Sigma_O = \{\leftarrow\}$

Blok:

$\Sigma_B = \{\{, \}\}$

Přiřazovací příkaz:

$\Sigma_P = \{<, -\}$

Výrazy (aritmetické, logické):

$\Sigma_V = \{+, -, *, /, ^, \&, |, !, <, >, =, (,)\}$

Klíčová slova:

$\Sigma_W = \{A..Z, a..z, 0..9\}$

Nyní všechny abecedy sloučíme. Ve všech těchto dílčích abecedách jsou vyjmenovány jen některé znaky, jiné zde přímo ve výčtu nejsou (@, \$ apod.), ale mohou se vyskytnout např. v rámci řetězců. Proto přibližně platí $\Sigma \sim \text{ASCII}$.

$\Sigma = \{ \text{TAB}, \leftarrow, \square, !, ", \#, \$, \%, \&, , (,), *, +, , -, ., /, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :, ;, <, =, >, ?, @, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, [, \,], ^, -, ', a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, \{, |, \}, \sim \}$

Tedy $\Sigma \sim \text{ASCII} (9, 13, 32 \dots 126)$.

2 Navržení lexikálního analyzátoru

2.1 Navržení jednotlivých dílčích gramatik

Nyní navrhujeme dílčí gramatiky pro jednotlivé prvky jazyka.

2.1.1 Nevýznamové znaky

Komentáře (K)

řádkový	# cokoli \leftarrow
víceřádkový	[cokoli]

$\Sigma_K = \{\Sigma\}$ (Pozn.: v komentáři může být jakýkoli znak.)

$N_K = \{K_0, K_1, K_2\}$

$G_K = (N_K, \Sigma_K, P_K, K_0)$

$P_K :$

$K_0 \rightarrow \#K_1|[K_2$

$K_1 \rightarrow (\Sigma - \{\leftrightarrow\})K_1| \leftarrow$

$K_2 \rightarrow (\Sigma - \{]\})K_2|]$

Bílé znaky (M)

$\Sigma_M = \{ \sqcup, \text{TAB} \}$

$N_M = \{M_0\}$

$G_M = (N_M, \Sigma_M, P_M, M_0)$

$P_M :$

$M_0 \rightarrow \sqcup M_0 | \text{TAB} M_0 | \sqcup | \text{TAB}$

2.1.2 Významové znaky

Identifikátory proměnných (I)

?cokoli?

$\Sigma_I = \{?, (\Sigma - \{?, \text{TAB}, \leftrightarrow\})\}$

$N_I = \{I_0, I_1\}$

$G_I = (N_I, \Sigma_I, P_I, I_0)$

$P_I :$

$I_0 \rightarrow ?I_1$

$I_1 \rightarrow (\Sigma - \{?, \sqcup, \text{TAB}, \leftrightarrow\})I_2$ (na první pozici nesmí být mezera)

$I_2 \rightarrow (\Sigma - \{?, \text{TAB}, \leftrightarrow\})I_2|?$

Číselné konstanty (N)

$\Sigma_N = \{+, -, , , 0..9\}$ (oddělovačem je desetinná čárka)

$N_N = \{N_0, N_1, N_2\}$

$G_N = (N_N, \Sigma_N, P_N, N_0)$

$P_N :$

$N_0 \rightarrow +N_1| - N_1| \check{c}N_1| \check{c}$

$N_1 \rightarrow \check{c}N_1| \check{c}|, N_2$

$N_2 \rightarrow \check{c}N_2| \check{c}$

Řetězcové konstanty (S)

$\Sigma_S = \Sigma$

$N_S = \{S_0, S_1, S_2\}$

$G_S = (N_S, \Sigma_S, P_S, S_0)$

$P_S :$

$S_0 \rightarrow "S_1$

$$S_1 \rightarrow (\Sigma - \{“, \backslash\})S_1|\backslash S_2| “$$

$$S_2 \rightarrow (\Sigma - \{“\})S_1|\backslash S_1| “S_1$$

Oddělovač příkazů (O)

$$\Sigma_O = \{\leftrightarrow\}$$

$$N_O = \{O_0\}$$

$$G_O = (N_O, \Sigma_O, P_O, O_0)$$

$$P_O :$$

$$O_0 \rightarrow \leftrightarrow$$

Blok (B)

$$\Sigma_B = \{\{, \}\}$$

$$N_B = \{B_0\}$$

$$G_B = (N_B, \Sigma_B, P_B, B_0)$$

$$P_B :$$

$$B_0 \rightarrow \{\}$$

Přiřazovací příkaz (P)

$$\Sigma_P = \{<, -\}$$

$$N_P = \{P_0, P_1\}$$

$$G_P = (N_P, \Sigma_P, P_P, P_0)$$

$$P_P :$$

$$P_0 \rightarrow < P_1$$

$$P_1 \rightarrow -$$

Výrazy a operátory (V)

$$\Sigma_V = \{+, -, *, /, ^, \&, |, !, <, >, =, (,)\}$$

$$N_V = \{V_0, V_4, V_5\}$$

$$G_V = (N_V, \Sigma_V, P_V, V_0)$$

$$P_V :$$

$$V_0 \rightarrow + | - | * | / | ^ | \& | | | ! | < | > | = | < V_4 | < V_5 | > V_4 | (|)$$

$$V_4 \rightarrow = (\text{symboly } <=, >=)$$

$$V_5 \rightarrow > (\text{symbol } <>)$$

Klíčová slova (W)

$$\Sigma_W = \{A..Z, a..z, 0..9\}$$

$$N_W = \{W_0, W_1\}$$

$$G_W = (N_W, \Sigma_W, P_W, W_0)$$

$$P_W :$$

$$W_0 \rightarrow A..ZW_1|A..Z$$

$$W_1 \rightarrow A..ZW_1|a..zW_1|0..9W_1|A..Z|a..z|0..9$$

2.2 Sestavení lexikálního analyzátoru, spojení dílčích gramatik

2.2.1 Nevýznamové jednotky (L_n)

Do nevýznamových znaků patří komentáře a bílé znaky. Aby bylo zajištěno, že tyto znaky se mohou opakovat, zařídíme, aby po terminálu, který ukončuje danou lexikální jednotku, mohla následovat jednotka další. Zavedeme tedy možnost „opakované výroby“ jednotek pomocí zopakování startovacího symbolu jazyka nevýznamových znaků – S_n . Gramatiky nevýznamových znaků budou vypadat takto:

$$\begin{aligned}
 S_n &\rightarrow K_0 \mid M_0 \\
 S_n &\rightarrow \#K_1 \mid [K_2 \mid \sqcup S_n \mid \text{TAB}S_n \mid \sqcup \mid \text{TAB} \\
 K_0 &\rightarrow \#K_1 \mid [K_2 \\
 K_1 &\rightarrow (\Sigma - \{\leftarrow\})K_1 \mid \leftarrow S_n \mid \leftarrow \\
 K_2 &\rightarrow (\Sigma - \{\mid\})K_2 \mid \mid S_n \mid \mid \\
 &(\text{Pravidlo } M_0 \rightarrow \sqcup M_0 \mid \text{TAB}M_0 \mid \sqcup \mid \text{TAB} \text{ bylo zintegrováno do pravidla } S_n.)
 \end{aligned}$$

2.2.2 Významové jednotky (L_v)

Do významových znaků řadíme identifikátory, čísla, řetězce, oddělovač příkazů, programový blok, přiřazovací příkaz, výrazy a klíčová slova. Jazyk významových jednotek se startovacím symbolem S_v má tuto podobu:

$$S_v \rightarrow I_0 \mid N_0 \mid S_0 \mid O_0 \mid B_0 \mid P_0 \mid V_0 \mid W_0$$

Jazyk nyní rozvineme a startovací symboly dílčích gramatik nahradíme pravou stranou přepisovacích pravidel:

$$S_v \rightarrow ?I_1 \mid +N_1 \mid -N_1 \mid \check{c}N_1 \mid \check{c} \mid "S_1 \mid \leftarrow \mid \{ \mid \} \mid < P_1 \mid + \mid - \mid * \mid / \mid ^ \mid \& \mid \mid \mid ! \mid < \mid > \mid = \mid < V_4 \mid < V_5 \mid > V_4 \mid (\mid) \mid A..Z \mid A..ZW_1$$

2.2.3 Sloučení významových a nevýznamových jednotek

Lexikální analyzátor má mít tuto podobu: (nevýznamový znak* významový znak)*, formálně zapsáno jako $S \rightarrow S_n S \mid S_v \mid \varepsilon$, což vyjadřuje, že může přijít několik nevýznamových lexikálů a jeden významový lexikál. Sdružíme-li významové a nevýznamové gramatiky dohromady, vznikne ucelená regulární gramatika se startovacím symbolem S .

$$S \rightarrow \#K_1 \mid [K_2 \mid \sqcup S \mid \text{TAB}S \mid \sqcup \mid \text{TAB} \mid S_v$$

Rozvineme-li startovací symbol jazyka významových znaků, S_v , dostaneme první pravidlo gramatiky jazyka iMPerator. Pravidla s indexem 0 jsou již obsažena v pravidle S , ostatní pravidla zůstávají shodná. Lexikální analyzátor pro náš jazyk vypadá následovně:

$$S \rightarrow \#K_1 \mid [K_2 \mid \sqcup S \mid \text{TAB}S \mid \sqcup \mid \text{TAB} \mid ?I_1 \mid + N_1 \mid - N_1 \mid \check{c}N_1 \mid \check{c} \mid \text{“}S_1 \mid \leftarrow \mid \{ \mid \} \mid < P_1 \mid + \mid - \mid * \mid / \mid ^ \mid \& \mid \mid \mid ! \mid < \mid > \mid = \mid < V_4 \mid < V_5 \mid > V_4 \mid (\mid) \mid A..Z \mid A..ZW_1$$

$$K_1 \rightarrow (\Sigma - \{\leftarrow\})K_1 \mid \leftarrow$$

$$K_2 \rightarrow (\Sigma - \{\}\})K_2 \mid]$$

$$I_1 \rightarrow (\Sigma - \{?, \sqcup \text{TAB}, \leftarrow\})I_2 \text{ (na první pozici nesmí být mezera)}$$

$$I_2 \rightarrow (\Sigma - \{?, \text{TAB}, \leftarrow\})I_2 \mid ?$$

$$N_1 \rightarrow \check{c}N_1 \mid \check{c} \mid N_2$$

$$N_2 \rightarrow \check{c}N_2 \mid \check{c}$$

$$S_1 \rightarrow (\Sigma - \{\text{“}, \backslash\})S_1 \mid \backslash S_2 \mid \text{“}$$

$$S_2 \rightarrow (\Sigma - \{\text{“}\})S_1 \mid \backslash S_1 \mid \text{“}S_1$$

$$P_1 \rightarrow -$$

$$V_4 \rightarrow =$$

$$V_5 \rightarrow >$$

$$W_1 \rightarrow A..ZW_1 \mid a..zW_1 \mid 0..9W_1 \mid A..Z \mid a..z \mid 0..9$$

2.3 NKA

Nedeterministický konečný automat

1. část	1	2	3	4	5	6	7	8	9	10	11
	#	[]	\sqcup	TAB	?	+	-	0..9	“	\leftarrow
S	K1	K2		S QF	S QF	I1	N1 QF	N1 QF	N1 QF	S1	QF
K1	K1	K1	K1	K1	K1	K1	K1	K1	K1	K1	S QF
K2	K2	K2	S QF	K2	K2	K2	K2	K2	K2	K2	K2
I1	I2	I2	I2				I2	I2	I2	I2	
I2	I2	I2	I2	I2		QF	I2	I2	I2	I2	
N1									N1 QF		
N2									N2 QF		
S1	S1	S1	S1	S1	S1	S1	S1	S1	S1	QF	S1
S2	S1	S1	S1	S1	S1	S1	S1	S1	S1	S1	S1
P1								QF			
V4											
V5											
W1									W1 QF		
QF											

2. část	12	13	14	15	16	17
	{,}	<	/, ^, &, —, !, (,)	=	*	>
S	QF	P1 V4 V5 QF	QF	QF	QF	QF V4
K1	K1	K1	K1	K1	K1	K1
K2	K2	K2	K2	K2	K2	K2
I1	I2	I2	I2	I2	I2	I2
I2	I2	I2	I2	I2	I2	I2
N1						
N2						
S1	S1	S1	S1	S1	S1	S1
S2	S1	S1	S1	S1	S1	S1
P1						
V4				QF		
V5						QF
W1						
QF						

3. část	18	19	20	21	22
	A..Z	,	\	a..z	\$, @, ., ;, :, %, -, ', ~
S	W1 QF				
K1	K1	K1	K1	K1	K1
K2	K2	K2	K2	K2	K2
I1	I2	I2	I2	I2	I2
I2	I2	I2	I2	I2	I2
N1		N2			
N2					
S1	S1	S1	S2	S1	S1
S2	S1	S1	S1	S1	S1
P1					
V4					
V5					
W1	W1 QF			W1 QF	
QF					

2.4 DKA

Deterministický konečný automat

1. část	1	2	3	4	5	6	7	8	9	10	11
	#	[]	␣	TAB	?	+	−	0..9	“	↔
K1	K1	K1	K1	K1	K1	K1	K1	K1	K1	K1	S QF
K2	K2	K2	S QF	K2	K2	K2	K2	K2	K2	K2	K2
I1	I2	I2	I2				I2	I2	I2	I2	
I2	I2	I2	I2	I2		QF	I2	I2	I2	I2	
N2									N2 QF		
S1	S1	S1	S1	S1	S1	S1	S1	S1	S1	QF	S1
S2	S1	S1	S1	S1	S1	S1	S1	S1	S1	S1	S1
← QF											
↔ SQF	K1	K2		SQF	SQF	I1	N1QF	N1QF	N1QF	S1	QF
← N1QF									N1QF		
← W1QF									W1QF		
← N2QF									N2QF		
← P1V4V5QF								QF			
← V4QF											

2. část	12	13	14	15	16	17
	{,}	<	/, :, %, ^, &, —, !, (,)	=	*	>
K1	K1	K1	K1	K1	K1	K1
K2	K2	K2	K2	K2	K2	K2
I1	I2	I2	I2	I2	I2	I2
I2	I2	I2	I2	I2	I2	I2
N2						
S1	S1	S1	S1	S1	S1	S1
S2	S1	S1	S1	S1	S1	S1
← QF						
↔ SQF	QF	P1V4V5QF	QF	QF	QF	V4QF
← N1QF						
← W1QF						
← N2QF						
← P1V4V5QF				QF		QF
← V4QF				QF		

3. část	18	19	20	21	22
	A..Z	,	\	a..z	\$, @, ., ;, -, ' , ~
K1	K1	K1	K1	K1	K1
K2	K2	K2	K2	K2	K2
I1	I2	I2	I2	I2	I2
I2	I2	I2	I2	I2	I2
N2					
S1	S1	S1	S2	S1	S1
S2	S1	S1	S1	S1	S1
← QF					
↔ SQF	W1QF				
← N1QF		N2			
← W1QF	W1QF			W1QF	
← N2QF					
← P1V4V5QF					
← V4QF					

3 Syntaktická analýza

3.1 Syntaktické diagramy

Prvním krokem při vytváření syntaktického analyzátoru bude nakreslení následujících syntaktických diagramů:

1. iMPerator (IM₀) – celý jazyk, obsahuje jen blok, deklarace nezavádíme;
2. blok (BL₀);
3. příkaz (PR₀) – jakýkoli povolený, může to být příkaz cyklu, rozhodování, definice funkce, volání funkce, vstupní a výstupní funkci, přiřazení a prázdný příkaz;
4. cyklus (CY₀);
5. rozhodování (RO₀);
6. definice funkce (FU₀);
7. volání funkce (VF₀);
8. vstup (IN₀);
9. výstup (OU₀);

10. přiřazení (PZ_0);
11. prázdný příkaz (PP_0);
12. podmínka (PO_0) – podmínka pro rozhodování;
13. formální parametry (FP_0) – parametry při definici funkce;
14. parametry funkce (PY_0) – parametry při volání funkce;
15. aritmetický výraz (AV_0);
16. relační výraz (RV_0).

Diagramy jsou uvedeny zvlášť jako příloha.

3.2 Gramatika pro syntaktickou analýzu

Syntaktické diagramy přepíšeme do gramatiky. Zde je uvedena první verze, která vznikla přímo ze syntaktických diagramů (bude dále upravena).

$$N_{iMP} = \{IM_0, BL_0, BL_1, PR_0, CY_0, RO_0, RO_1, FU_0, FU_1, VF_0, VF_1, IN_0, OU_0, PZ_0, PP_0, PO_0, FP_0, FP_1, PY_0, PY_1, AV_0, V_1, V_2, V_3, RV_0, R_1, R_2, R_3\}$$

$$G_{iMP} = (N_{iMP}, \Sigma_{iMP}, P_{iMP}, IM_0)$$

$$\begin{aligned} IM_0 &\rightarrow BL_0 \\ BL_0 &\rightarrow \{PR_0\} | \{PR_0 BL_1 \\ BL_1 &\rightarrow \leftarrow PR_0 BL_1 \\ PR_0 &\rightarrow CY_0 | RO_0 | FU_0 | VF_0 | IN_0 | OU_0 | PZ_0 | PP_0 \\ CY_0 &\rightarrow \underline{DONEC} PO_0 BL_0 \\ RO_0 &\rightarrow \underline{SI} PO_0 \underline{ERGO} BL_0 RO_1 \\ RO_1 &\rightarrow \underline{CETERA} BL_0 \underline{IS} | \underline{IS} \\ FU_0 &\rightarrow \underline{FUNCTIO} \underline{názevfunkce} (FU_1 \\ FU_1 &\rightarrow FP_0) BL_0 |) BL_0 \\ VF_0 &\rightarrow \underline{názevfunkce} (VF_1 \\ VF_1 &\rightarrow PY_0) |) \\ IN_0 &\rightarrow \underline{LECTO} (\underline{identif}) \\ OU_0 &\rightarrow \underline{SCRIBO} (RV_0) \\ PZ_0 &\rightarrow \underline{identif} < -AV_0 | \underline{identif} < -RV_0 \\ PP_0 &\rightarrow \varepsilon \\ PO_0 &\rightarrow (RV_0) \\ FP_0 &\rightarrow \underline{identif} FP_1 | \underline{identif} \\ FP_1 &\rightarrow ; \underline{identif} FP_1 | ; \underline{identif} \end{aligned}$$

$$\begin{aligned}
PY_0 &\rightarrow RV_0 PY_1 \mid RV_0 \\
PY_1 &\rightarrow ; RV_0 PY_1 \mid ; RV_0 \\
AV_0 &\rightarrow AV_0 + V_1 \mid AV_0 - V_1 \mid V_1 \\
V_1 &\rightarrow V_1 * V_2 \mid V_1 / V_2 \mid V_2 \\
V_2 &\rightarrow V_2 \wedge V_3 \mid V_3 \\
V_3 &\rightarrow (RV_0) \mid \text{\texttt{řetězec}} \mid \text{\texttt{číslo}} \mid \text{\texttt{identif}} \mid \text{\texttt{názevfunkce}} \\
RV_0 &\rightarrow RV_0 \& R_1 \mid RV_0 \mid R_1 \mid R_1 \\
R_1 &\rightarrow !R_2 \mid R_2 \\
R_2 &\rightarrow AV_0 \text{\texttt{ro}} AV_0 \mid AV_0
\end{aligned}$$

Pozn.: identif je identifikátor proměnné.

Takto vzniklá gramatika se samozřejmě musí upravit. Bude třeba upravit pravé strany pravidel a odstranit bezprostřední levou rekurzi a připravit tak pravidla pro výpočet množin First a Follow. Zmizí také pravidlo $PP_0 \rightarrow \varepsilon$; bude zintegrováno do pravidla, které řeší příkaz. Dále je potřeba odstranit společný výskyt neterminálů AV_0 a RV_0 v jednom pravidle. Výsledná úprava je pak následující:

$$\begin{aligned}
N_{iMP} = \{ &IM_0, BL_0, BL_1, PR_0, CY_0, RO_0, RO_1, FU_0, FU_1, VF_0, VF_1, IN_0, \\
&OU_0, PZ_0, PO_0, FP_0, FP_1, PY_0, PY_1, AV_0, \overline{AV}_0, V_1, \overline{V}_1, V_2, \overline{V}_2, V_3, \\
&RV_0, \overline{RV}_0, R_1, R_2, R_3 \}
\end{aligned}$$

$$G_{iMP} = (N_{iMP}, \Sigma_{iMP}, P_{iMP}, IM_0)$$

$$\begin{aligned}
IM_0 &\rightarrow BL_0 \\
BL_0 &\rightarrow \{ PR_0 BL_1 \\
BL_1 &\rightarrow \leftrightarrow PR_0 BL_1 \mid \} \\
PR_0 &\rightarrow CY_0 \mid RO_0 \mid FU_0 \mid VF_0 \mid IN_0 \mid OU_0 \mid PZ_0 \mid \varepsilon \\
CY_0 &\rightarrow \text{\texttt{DONEC}} PO_0 BL_0 \\
RO_0 &\rightarrow \text{\texttt{SI}} PO_0 \text{\texttt{ERGO}} BL_0 RO_1 \\
RO_1 &\rightarrow \text{\texttt{CETERA}} BL_0 \text{\texttt{IS}} \mid \text{\texttt{IS}} \\
FU_0 &\rightarrow \text{\texttt{FUNCTIO}} \text{\texttt{názevfunkce}} (FU_1 \\
FU_1 &\rightarrow FP_0) BL_0 \mid) BL_0 \\
VF_0 &\rightarrow \text{\texttt{názevfunkce}} (VF_1 \\
VF_1 &\rightarrow PY_0) \mid) \\
IN_1 &\rightarrow \text{\texttt{LECTO}} (\text{\texttt{identif}}) \\
OU_0 &\rightarrow \text{\texttt{SCRIBO}} (RV_0) \\
PZ_0 &\rightarrow \text{\texttt{identif}} < -RV_0 \\
PO_0 &\rightarrow (RV_0) \\
FP_0 &\rightarrow \text{\texttt{identif}} FP_1 \\
FP_1 &\rightarrow ; \text{\texttt{identif}} FP_1 \mid \varepsilon \\
PY_0 &\rightarrow RV_0 PY_1 \\
PY_1 &\rightarrow ; RV_0 PY_1 \mid \varepsilon
\end{aligned}$$

$$\begin{aligned}
& AV_0 \rightarrow V_1 \overline{AV_0} \\
& \overline{AV_0} \rightarrow +V_1 \overline{AV_0} \mid -V_1 \overline{AV_0} \mid \varepsilon \\
& V_1 \rightarrow V_2 \overline{V_1} \\
& \overline{V_1} \rightarrow *V_2 \overline{V_1} \mid /V_2 \overline{V_1} \mid \varepsilon \\
& V_2 \rightarrow V_3 \overline{V_2} \\
& \overline{V_2} \rightarrow ^V_3 \overline{V_2} \mid \varepsilon \\
& V_3 \rightarrow (RV_0) \mid \underline{\text{řetězec}} \mid \underline{\text{číslo}} \mid \underline{\text{identif}} \mid \underline{\text{názevfunkce}} \\
& RV_0 \rightarrow R_1 \overline{RV_0} \\
& \overline{RV_0} \rightarrow \&R_1 \overline{RV_0} \mid |R_1 \overline{RV_0} \mid \varepsilon \\
& R_1 \rightarrow !R_2 \mid R_2 \\
& R_2 \rightarrow AV_0 R_3 \\
& R_3 \rightarrow \underline{\text{ro}} AV_0 \mid \varepsilon
\end{aligned}$$

3.3 Výpočet množin First a Follow, ověření LL(1)

Dalším krokem bude výpočet množin First a Follow a ověření toho, zda se jedná o LL(1) jazyk a gramatiku. Výsledkem je, že jazyk je skutečně LL(1).

$$\begin{aligned}
& \text{FF} (IM_0 \rightarrow BL_0) = \text{FI} (IM_0 \rightarrow BL_0) = \{ \{ \} \\
& \text{FF} (BL_0 \rightarrow \{PR_0 BL_1\}) = \text{FI} (BL_0 \rightarrow \{PR_0 BL_1\}) = \{ \{ \} \\
& \text{FF} (BL_1 \rightarrow \leftrightarrow PR_0 BL_1) = \text{FI} (BL_1 \rightarrow \leftrightarrow PR_0 BL_1) = \{ \leftrightarrow \} \\
& \text{FF} (BL_1 \rightarrow \}) = \text{FI} (BL_1 \rightarrow \}) = \{ \} \} \\
& \{ \} \} \cap \{ \leftrightarrow \} = \emptyset \\
& \text{FF} (PR_0 \rightarrow CY_0) = \text{FI} (PR_0 \rightarrow CY_0) = \{ \underline{\text{DONEC}} \} \\
& \text{FF} (PR_0 \rightarrow RO_0) = \text{FI} (PR_0 \rightarrow RO_0) = \{ \underline{\text{SI}} \} \\
& \text{FF} (PR_0 \rightarrow FU_0) = \text{FI} (PR_0 \rightarrow FU_0) = \{ \underline{\text{FUNCTIO}} \} \\
& \text{FF} (PR_0 \rightarrow VF_0) = \text{FI} (PR_0 \rightarrow VF_0) = \{ \underline{\text{názevfunkce}} \} \\
& \text{FF} (PR_0 \rightarrow IN_0) = \text{FI} (PR_0 \rightarrow IN_0) = \{ \underline{\text{LECTO}} \} \\
& \text{FF} (PR_0 \rightarrow OU_0) = \text{FI} (PR_0 \rightarrow OU_0) = \{ \underline{\text{SCRIBO}} \} \\
& \text{FF} (PR_0 \rightarrow PZ_0) = \text{FI} (PR_0 \rightarrow PZ_0) = \{ \underline{\text{identif}} \} \\
& \text{FF} (PR_0 \rightarrow \varepsilon) = \text{FO} (PR_0) = \{ \leftrightarrow, \} \} \\
& \{ \underline{\text{DONEC}} \} \cap \{ \underline{\text{SI}} \} \cap \{ \underline{\text{FUNCTIO}} \} \cap \{ \underline{\text{názevfunkce}} \} \cap \{ \underline{\text{LECTO}} \} \cap \{ \underline{\text{SCRIBO}} \} \\
& \cap \{ \underline{\text{identif}} \} \cap \{ \leftrightarrow, \} \} = \emptyset \\
& \text{FF} (CY_0 \rightarrow \underline{\text{DONEC}} PO_0 BL_0) = \text{FI} (CY_0 \rightarrow \underline{\text{DONEC}} PO_0 BL_0) = \{ \underline{\text{DONEC}} \} \\
& \text{FF} (RO_0 \rightarrow \underline{\text{SI}} PO_0 \underline{\text{ERGO}} BL_0 RO_1) = \text{FI} (RO_0 \rightarrow \underline{\text{SI}} PO_0 \underline{\text{ERGO}} BL_0 RO_1) \\
& = \{ \underline{\text{SI}} \} \\
& \text{FF} (RO_1 \rightarrow \underline{\text{CETERA}} BL_0 \underline{\text{IS}}) = \text{FI} (RO_1 \rightarrow \underline{\text{CETERA}} BL_0 \underline{\text{IS}}) = \{ \underline{\text{CETERA}} \} \\
& \text{FF} (RO_1 \rightarrow \underline{\text{IS}}) = \text{FI} (RO_1 \rightarrow \underline{\text{IS}}) = \{ \underline{\text{IS}} \} \\
& \{ \underline{\text{CETERA}} \} \cap \{ \underline{\text{IS}} \} = \emptyset
\end{aligned}$$

$$\text{FF} (FU_0 \rightarrow \text{FUNCTIO } \underline{\text{názevfunkce}} (FU_1) = \text{FI} (FU_0 \rightarrow \text{FUNCTIO } \underline{\text{názevfunkce}} (FU_1) \\ = \{\text{FUNCTIO}\}$$

$$\text{FF} (FU_1 \rightarrow FP_0)BL_0 = \text{FI} (FU_1 \rightarrow FP_0)BL_0 = \{\underline{\text{identif}}\}$$

$$\text{FF} (FU_1 \rightarrow)BL_0 = \text{FI} (FU_1 \rightarrow)BL_0 = \{ \} \}$$

$$\{\underline{\text{identif}}\} \cap \{ \} = \emptyset$$

$$\text{FF} (VF_0 \rightarrow \underline{\text{názevfunkce}}(VF_1) = \text{FI} (VF_0 \rightarrow \underline{\text{názevfunkce}}(VF_1) = \{\underline{\text{názevfunkce}}\}$$

$$\text{FF} (VF_1 \rightarrow PY_0)) = \text{FI} (VF_1 \rightarrow PY_0)) = \{ !, (, \underline{\text{řetězec}}, \underline{\text{číslo}}, \underline{\text{identif}}, \\ \underline{\text{názevfunkce}}\}$$

$$\text{FF} (VF_1 \rightarrow)) = \text{FI} (VF_1 \rightarrow)) = \{ \} \}$$

$$\{ !, (, \underline{\text{řetězec}}, \underline{\text{číslo}}, \underline{\text{identif}}, \underline{\text{názevfunkce}}\} \cap \{ \} = \emptyset$$

$$\text{FF} (IN_0 \rightarrow \text{LECTO} (\underline{\text{identif}})) = \text{FI} (IN_0 \rightarrow \text{LECTO} (\underline{\text{identif}})) = \{\text{LECTO}\}$$

$$\text{FF} (OU_0 \rightarrow \text{SCRIBO} (RV_0)) = \text{FI} (OU_0 \rightarrow \text{SCRIBO} (RV_0)) = \{\text{SCRIBO}\}$$

$$\text{FF} (PZ_0 \rightarrow \underline{\text{identif}} < -RV_0) = \text{FI} (PZ_0 \rightarrow \underline{\text{identif}} < -RV_0) = \{\underline{\text{identif}}\}$$

$$\text{FF} (PO_0 \rightarrow (RV_0)) = \text{FI} (PO_0 \rightarrow (RV_0)) = \{ (\}$$

$$\text{FF} (FP_0 \rightarrow \underline{\text{identif}} FP_1) = \text{FI} (FP_0 \rightarrow \underline{\text{identif}} FP_1) = \{\underline{\text{identif}}\}$$

$$\text{FF} (FP_1 \rightarrow ; \underline{\text{identif}} FP_1) = \text{FI} (FP_1 \rightarrow ; \underline{\text{identif}} FP_1) = \{ ; \}$$

$$\text{FF} (FP_1 \rightarrow \varepsilon) = \text{FO} (FP_1) = \{ \} \}$$

$$\{ ; \} \cap \{ \} = \emptyset$$

$$\text{FF} (PY_0 \rightarrow RV_0 PY_1) = \text{FI} (PY_0 \rightarrow RV_0 PY_1) = \{ !, (, \underline{\text{řetězec}}, \underline{\text{číslo}}, \\ \underline{\text{identif}}, \underline{\text{názevfunkce}}\}$$

$$\text{FF} (PY_1 \rightarrow ; RV_0 PY_1) = \text{FI} (PY_1 \rightarrow ; RV_0 PY_1) = \{ ; \}$$

$$\text{FF} (PY_1 \rightarrow \varepsilon) = \text{FO} (PY_1) = \{ \} \}$$

$$\{ ; \} \cap \{ \} = \emptyset$$

$$\text{FF} (AV_0 \rightarrow V_1 \overline{AV_0}) = \text{FI} (AV_0 \rightarrow V_1 \overline{AV_0}) = \{ (, \underline{\text{řetězec}}, \underline{\text{číslo}}, \underline{\text{identif}}, \\ \underline{\text{názevfunkce}}\}$$

$$\text{FF} (\overline{AV_0} \rightarrow + V_1 \overline{AV_0}) = \text{FI} (\overline{AV_0} \rightarrow + V_1 \overline{AV_0}) = \{ + \}$$

$$\text{FF} (\overline{AV_0} \rightarrow - V_1 \overline{AV_0}) = \text{FI} (\overline{AV_0} \rightarrow - V_1 \overline{AV_0}) = \{ - \}$$

$$\text{FF} (\overline{AV_0} \rightarrow \varepsilon) = \text{FO} (\overline{AV_0}) = \{\underline{\text{ro}}, \&, |, \leftarrow,), \}, ; \}$$

$$\{ + \} \cap \{ - \} \cap \{\underline{\text{ro}}, \&, |, \leftarrow,), \}, ; \} = \emptyset$$

$$\text{FF} (V_1 \rightarrow V_2 \overline{V_1}) = \text{FI} (V_1 \rightarrow V_2 \overline{V_1}) = \{ (, \underline{\text{řetězec}}, \underline{\text{číslo}}, \underline{\text{identif}}, \underline{\text{názevfunkce}}\}$$

$$\text{FF} (\overline{V_1} \rightarrow * V_2 \overline{V_1}) = \text{FI} (\overline{V_1} \rightarrow * V_2 \overline{V_1}) = \{ * \}$$

$$\text{FF} (\overline{V_1} \rightarrow / V_2 \overline{V_1}) = \text{FI} (\overline{V_1} \rightarrow / V_2 \overline{V_1}) = \{ / \}$$

$$\text{FF} (\overline{V_1} \rightarrow \varepsilon) = \text{FO} (\overline{V_1}) = \{ +, -, \underline{\text{ro}}, \&, |, \leftarrow,), \}, ; \}$$

$$\{ * \} \cap \{ / \} \cap \{ +, -, \underline{\text{ro}}, \&, |, \leftarrow,), \}, ; \} = \emptyset$$

$$\text{FF} (V_2 \rightarrow V_3 \overline{V_2}) = \text{FI} (V_2 \rightarrow V_3 \overline{V_2}) = \{ (, \underline{\text{řetězec}}, \underline{\text{číslo}}, \underline{\text{identif}}, \underline{\text{názevfunkce}} \}$$

$$\text{FF} (\overline{V_2} \rightarrow \wedge V_3 \overline{V_2}) = \text{FI} (\overline{V_2} \rightarrow \wedge V_3 \overline{V_2}) = \{ \wedge \}$$

$$\text{FF} (\overline{V_2} \rightarrow \varepsilon) = \text{FO} (\overline{V_2}) = \{ *, /, +, -, \underline{\text{ro}}, \&, |, \leftrightarrow,), \}, ; \}$$

$$\{ \wedge \} \cap \{ *, /, +, -, \underline{\text{ro}}, \&, |, \leftrightarrow,), \}, ; \} = \emptyset$$

$$\text{FF} (V_3 \rightarrow (RV_0)) = \text{FI} (V_3 \rightarrow (RV_0)) = \{ (\}$$

$$\text{FF} (V_3 \rightarrow \underline{\text{řetězec}}) = \text{FI} (V_3 \rightarrow \underline{\text{řetězec}}) = \{ \underline{\text{řetězec}} \}$$

$$\text{FF} (V_3 \rightarrow \underline{\text{číslo}}) = \text{FI} (V_3 \rightarrow \underline{\text{číslo}}) = \{ \underline{\text{číslo}} \}$$

$$\text{FF} (V_3 \rightarrow \underline{\text{identif}}) = \text{FI} (V_3 \rightarrow \underline{\text{identif}}) = \{ \underline{\text{identif}} \}$$

$$\text{FF} (V_3 \rightarrow VF_0) = \text{FI} (V_3 \rightarrow VF_0) = \{ \underline{\text{názevfunkce}} \}$$

$$\{ (\} \cap \{ \underline{\text{řetězec}} \} \cap \{ \underline{\text{číslo}} \} \cap \{ \underline{\text{identif}} \} \cap \{ \underline{\text{názevfunkce}} \} = \emptyset$$

$$\text{FF} (RV_0 \rightarrow R_1 \overline{RV_0}) = \text{FI} (RV_0 \rightarrow R_1 \overline{RV_0}) = \{ !, (, \underline{\text{řetězec}}, \underline{\text{číslo}}, \underline{\text{identif}}, \underline{\text{názevfunkce}} \}$$

$$\text{FF} (\overline{RV_0} \rightarrow \& R_1 \overline{RV_0}) = \text{FI} (\overline{RV_0} \rightarrow \& R_1 \overline{RV_0}) = \{ \& \}$$

$$\text{FF} (\overline{RV_0} \rightarrow | R_1 \overline{RV_0}) = \text{FI} (\overline{RV_0} \rightarrow | R_1 \overline{RV_0}) = \{ | \}$$

$$\text{FF} (\overline{RV_0} \rightarrow \varepsilon) = \text{FO} (\overline{RV_0}) = \{), ; , \leftrightarrow, \} \}$$

$$\{ \& \} \cap \{ | \} \cap \{), ; , \leftrightarrow, \} \} = \emptyset$$

$$\text{FF} (R_1 \rightarrow !R_2) = \text{FI} (R_1 \rightarrow !R_2) = \{ ! \}$$

$$\text{FF} (R_1 \rightarrow R_2) = \text{FI} (R_1 \rightarrow R_2) = \text{FI} (R_1 \rightarrow AV_0 R_3) = \{ (, \underline{\text{řetězec}}, \underline{\text{číslo}}, \underline{\text{identif}}, \underline{\text{názevfunkce}} \}$$

$$\{ ! \} \cap \{ (, \underline{\text{řetězec}}, \underline{\text{číslo}}, \underline{\text{identif}}, \underline{\text{názevfunkce}} \} = \emptyset$$

$$\text{FF} (R_2 \rightarrow AV_0 R_3) = \text{FI} (R_2 \rightarrow AV_0 R_3) = \{ (, \underline{\text{řetězec}}, \underline{\text{číslo}}, \underline{\text{identif}}, \underline{\text{názevfunkce}} \}$$

$$\text{FF} (R_3 \rightarrow \underline{\text{ro}} AV_0) = \text{FI} (R_3 \rightarrow \underline{\text{ro}} AV_0) = \{ \underline{\text{ro}} \}$$

$$\text{FF} (R_3 \rightarrow \varepsilon) = \text{FO} (R_3) = \{ \&, |, \leftrightarrow,), \}, ; \}$$

$$\{ \underline{\text{ro}} \} \cap \{ \&, |, \leftrightarrow,), \}, ; \} = \emptyset$$

3.4 Rozkladová tabulka

1. část	1	2	3	4	5	6
	{	}	\leftrightarrow	<u>DONEC</u>	<u>SI</u>	<u>IS</u>
IM_0	BL_0					
BL_0	$\{PR_0BL_1$					
BL_1		}	$\leftrightarrow PR_0BL_1$			
PR_0		ε	ε	CY_0	RO_0	
CY_0				<u>DONEC</u> PO_0BL_0		
RO_0					<u>SI</u> PO_0 <u>ERGO</u> BL_0RO_1	
RO_1						<u>IS</u>
FU_0						
FU_1						
VF_0						
VF_1						
IN_0						
OU_0						
PZ_0						
PO_0						
FP_0						
FP_1						
PY_0						
PY_1						
AV_0						
\overline{AV}_0		ε	ε			
V_1						
\overline{V}_1		ε	ε			
V_2						
\overline{V}_2		ε	ε			
V_3						
RV_0						
\overline{RV}_0		ε	ε			
R_1						
R_2						
R_3		ε	ε			

2. část	7	8	9	10
	<u>CETERA</u>	<u>FUNCTIO</u>	<u>LECTO</u>	<u>SCRIBO</u>
IM_0				
BL_0				
BL_1				
PR_0		FU_0	IN_0	OU_0
CY_0				
RO_0				
RO_1	<u>CETERA</u> BL_0 <u>IS</u>			
FU_0		<u>FUNCTIO</u> názevfunkce (FU_1		
FU_1				
VF_0				
VF_1				
IN_0			<u>LECTO</u> (identif)	
OU_0				<u>SCRIBO</u> (RV_0)
PZ_0				
PO_0				
FP_0				
FP_1				
PY_0				
PY_1				
AV_0				
AV_0				
V_1				
\bar{V}_1				
V_2				
\bar{V}_2				
V_3				
RV_0				
\bar{RV}_0				
R_1				
R_2				
R_3				

3. část	11	12	13	14	15	16
	<u>řetězec</u>	<u>číslo</u>	<u>identif</u>	<u>názevfunkce</u>	()
IM_0						
BL_0						
BL_1						
PR_0			PZ_0	VF_0		
CY_0						
RO_0						
RO_1						
FU_0						
FU_1			$FP_0)BL_0$			$)BL_0$
VF_0				<u>názevfunkce</u> (VF_1		
VF_1	$PY_0)$	$PY_0)$	$PY_0)$	$PY_0)$	$PY_0)$	$)$
IN_0						
OU_0						
PZ_0			<u>identif</u> $< -RV_0$			
PO_0					(RV_0)	
FP_0			<u>identif</u> FP_1			
FP_1						ε
PY_0	RV_0PY_1	RV_0PY_1	RV_0PY_1	RV_0PY_1	RV_0PY_1	
PY_1						ε
AV_0	$V_1\overline{AV_0}$	$V_1\overline{AV_0}$	$V_1\overline{AV_0}$	$V_1\overline{AV_0}$	$V_1\overline{AV_0}$	
$\overline{AV_0}$						ε
V_1	$V_2\overline{V_1}$	$V_2\overline{V_1}$	$V_2\overline{V_1}$	$V_2\overline{V_1}$	$V_2\overline{V_1}$	
$\overline{V_1}$						ε
V_2	$V_3\overline{V_2}$	$V_3\overline{V_2}$	$V_3\overline{V_2}$	$V_3\overline{V_2}$	$V_3\overline{V_2}$	
$\overline{V_2}$						ε
V_3	<u>řetězec</u>	<u>číslo</u>	<u>identif</u>	VF_0	(RV_0)	
RV_0	$R_1\overline{RV_0}$	$R_1\overline{RV_0}$	$R_1\overline{RV_0}$	$R_1\overline{RV_0}$	$R_1\overline{RV_0}$	
$\overline{RV_0}$						ε
R_1	R_2	R_2	R_2	R_2	R_2	
R_2	AV_0R_3	AV_0R_3	AV_0R_3	AV_0R_3	AV_0R_3	
R_3						ε

4. část	17	18	19	20	21
	;	+	−	*	/
IM_0					
BL_0					
BL_1					
PR_0					
CY_0					
RO_0					
RO_1					
FU_0					
FU_1					
VF_0					
VF_1					
IN_0					
OU_0					
PZ_0					
PO_0					
FP_0					
FP_1	;identif FP_1				
PY_0					
PY_1	; RV_0PY_1				
AV_0					
AV_0	ε	$+V_1\overline{AV_0}$	$+V_1\overline{AV_0}$		
V_1					
$\overline{V_1}$	ε	ε	ε	$*V_2\overline{V_1}$	$/V_2\overline{V_1}$
V_2					
$\overline{V_2}$	ε	ε	ε	ε	ε
V_3					
RV_0					
$\overline{RV_0}$	ε				
R_1					
R_2					
R_3	ε				

5. část	22	23	24	25	26	27
	\wedge	$\&$	$ $	$!$	$\underline{\text{ro}}$	ε
IM_0						
BL_0						
BL_1						
PR_0						
CY_0						
RO_0						
RO_1						
FU_0						
FU_1						
VF_0						
VF_1				$PY_0)$		
IN_0						
OU_0						
PZ_0						
PO_0						
FP_0						
FP_1						
PY_0				RV_0PY_1		
PY_1						
AV_0						
AV_0		ε	ε		ε	
V_1						
$\overline{V_1}$		ε	ε		ε	
V_2						
$\overline{V_2}$	$\wedge V_3 \overline{V_2}$	ε	ε		ε	
V_3						
RV_0				$R_1 \overline{RV_0}$		
$\overline{RV_0}$		$\& R_1 \overline{RV_0}$	$ R_1 \overline{RV_0}$			
R_1				$!R_2$		
R_2						
R_3		ε	ε		$\underline{\text{ro}} AV_0$	

4 Jak programovat v jazyku iMPerator

Programovací jazyk iMPerator má svá určitá specifika, kterými se odlišuje od jiných programovacích jazyků. Zde jsou uvedeny dvě nejdůležitější:

- Bloky. Celý program je uzavřen v jednom „globálním“ bloku, tzn. začíná a končí složenou závorkou. Bloky je možno mít pouze tam, kde je to povoleno (viz SYDy).
- Jednotlivé příkazy se od sebe oddělují znakem prázdného řádku, nikoli středníkem (středník je naopak použit jako oddělovač parametrů funkce). Jestliže příkaz obsahuje blok, musí být složená závorka, která tento blok uvozuje, uvedena na stejném řádku jako předchozí klíčové slovo či podmínka. Znak konce řádku se považuje za nový příkaz a předchozí příkaz (typicky DONEC, SI) by nebyl ukončen.
- Jazyk je netypový, neuvádějí se žádné deklarace a definice. Pouze definice funkcí musí být uvedena před prvním voláním, ačkoli to nutně nemusí být na úplném začátku programu.
- Klíčová slova se píší velkými písmeny. (Zde opět poukazujeme na odkaz Římanů.)
- Identifikátor proměnné je uzavřen mezi dva otazníky.
- Celá a desetinná část čísla se odděluje desetinnou čárkou.
- Řetězec může obsahovat znak konce řádku.
- Symbol pro přiřazení není ani =, ani :=, nýbrž <-.

Dále se v jazyce iMPerator programuje běžným způsobem. Syntaxe je nejlépe patrná ze syntaktických diagramů. Pouze je potřeba mít na paměti zatím omezené prostředky jazyka:

- Existuje pouze cyklus s podmínkou na začátku (DONEC).
- Funkce SCRIBO vypisuje buď jen zadanou hodnotu (řetězce, čísla) nebo hodnotu k vypočítání (identifikátory). Sekvenci, která obsahuje oba typy vypisovaných údajů, je potřeba rozdělit na několik příkazů.
- Jazyk zvládá standardní operace sčítání, násobení, odčítání, dělení, umocňování a logického and, or a negace (s prioritou). Nejsou implementovány například užitečné operátory ++ nebo --.

To však nebrání používání jazyka, který je i s těmito prostředky funkční.

Poznámka: Všechny programy uvedené v příloze obsahují na konci příkaz pro výpis řetězce „AHOJ!“. Rozhodně to není nutnost, programy jsou pouze slušně vychované a při odchodu pozdraví.

5 Implementace

Datové struktury a jejich obsluha

Za základní paměťovou strukturu jsme zvolili obousměrný zřetězený dynamický seznam. Tato konstrukce nám díky své obecnosti dovoluje elegantně implementovat jak fronty, tak zásobníky, případně je vhodně kombinovat.

Základní funkce pro přidávání prvků do seznamu jsou `PridejNaKonec` a `PridejNaZacatek`. Destruktivní čtení (se zrušením prvku seznamu) zajišťují funkce `NactiOdZacatku`, `NactiOdKonce`. Nedestruktivní čtení provádějí funkce `CtiOdZacatku`, `CtiOdKonce`.

V programu používáme tyto seznamy:

- **prom** – seznam používaných proměnných, datovou složkou je záznam obsahující název proměnné, její obsah a hodnotu scope. S tímto seznamem pracují funkce `PridejPromennou` a `CtiPromennou`, jejichž názvy dostatečně vypovídají o funkci, a funkce `ZrusScope`, která se volá vždy po ukončení těla funkce a odstraňuje z paměti všechny lokální proměnné.
- **kolej**, **postfix**, **vypocet** – seznamy jsou nutné pro vyhodnocování výrazů. Zásobník **kolej** slouží jako slepá kolej, pro vytvoření postfixového výrazu složí fronta **postfix**, kam se podle potřeby posílají prvky buď z původního výrazu, nebo z koleje. Hodnota postfixového výrazu se vyčíslí pomocí zásobníku **vypocet**. Drobný problém vznikl při zanořování funkcí, kdy na kolejích zůstává ještě nevyhodnocený výraz a ve funkci se vyhodnocují mezitím jiné výrazy. Tento problém jsme vyřešili speciální zarážkou na koleji, která se přidá vždy při volání funkce ve výrazu a odstraní až po jejím dokončení. Jinak vyhodnocování výrazů probíhá standardně metodou slepé koleje.
- **funkce** – skládá se z prvků, které obsahují vždy název dané funkce a odkaz na tělo funkce. Tělo je tvořeno samostatným zřetězeným seznamem. Seznam začíná formálními parametry funkce, po nich následuje znak pravé kulaté závorky (ukončení parametrů) a dále je vlastní blok těla funkce. Se seznamem pracují funkce `PridejFunkci` (volá se, pokud interpret narazí na deklaraci funkce; zabezpečí uložení jejího těla, aby

mohla být později volána) a `NajdiFunkci` (najde existující funkci v seznamu a vrátí ukazatel na její tělo).

- `lexroua` – obsahuje kód, který se bude přednostně zpracovávat funkcí `lex`. Ukládá se do něj přednačtená lexikální jednotka z minulého průchodu lexu a pak je také plněn kódem při nelineární struktuře programu (cyklus, volání funkce, ...).
- `program` – pomocný seznam, do kterého se zaznamenává kód programu, pokud je příznak `ZaznamProgramu` nastaven na hodnotu 1 (používá se třeba při záznamu těla cyklu).

Důležité funkce

Základní funkcí je funkce `lex`, která kontroluje lexikální správnost kódu, jejím výstupem je zpracovaný kód, obsah tokenu a jeho typ. Vstupem je funkce `CtiZnak`, která předává i znak ze seznamu `lexroua` nebo ze vstupního souboru. Pokud je nastaven příznak `ZaznamProgramu`, je před skončením uložena kopie zpracovaného kódu do seznamu `program`.

Pro vyčíslování hodnoty aritmeticko-logických výrazů slouží funkce `arita`, `priorita`, `vypocti`, `Vlacek`, `ZpracujPrvek`. Funkci `ZpracujPrvek` se posílají jednotlivé prvky výrazu a funkce je dle příslušných pravidel ukládá ve správném pořadí (dle priority) na správnou kolej. Po odeslání všech prvků výrazu se spustí funkce `Vlacek`, která dokončí převod do postfixového tvaru (konkrétně vysype zbylé operátory z koleje na konec postfixu) a provede pomocí funkce `vypocet` výpočet. `Arita` a `priorita` jednotlivých operátorů jsou nastaveny ve stejnojmenných funkcích.

Ostatní funkce jsou součástí sémantického analyzátoru a odpovídají rozkladové tabulce.

V hlavní funkci `main` je jen otestován počet parametrů; pokud je jako parametr soubor, začne se provádět, jinak se očekává program pro interpretaci na standardním vstupu.

Programátorské poznatky

Při vytváření interpretu bylo nejpracnější odladění sémantických akcí, stále se vyskytovaly nějaké problémy s ukazateli, nealokovanou nebo neodalokovanou pamětí či použitím neinicializované proměnné. Tyto problémy byly dány tím, že autoři tohoto projektu nemají příliš mnoho praktických zkušeností s dynamickými strukturami ani s jazykem C++. Současně se domnívají, že nutnost vytvářet na vše vlastní obslužné programky a neustále hlídat ukazatele je programátorsky nepříjemné. Z těchto důvodů bylo nejzapeklitější navr-

žení zpracování funkcí tak, aby fungovaly i rekurzivně. Dále autoři vyslovují výhrady ke způsobu práce s řetězcí, která je v jazyce C velice neohrabaná.

6 Závěr

Cílem této práce bylo sestavit interpret vlastního jazyka. V závěru práce můžeme konstatovat, že cíle bylo dosaženo; světlo světa spatřil nový interpret jazyka iMPerator. Programátorské prostředky jsou sice omezené, ale interpret je funkční a je v něm možno psát jednodušší programy. Přejeme našemu jazyku úspěšný rozvoj.

7 Příloha – testovací soubory

Příloha obsahuje zdrojové kódy šesti testovacích programů v jazyce iMPerator. Vysvětlivky jsou obsaženy v kapitole „Jak programovat“.

7.1 Kalkulačka

Program si vyžádá první operátor, operand (slovy) a druhý operátor. Prove s nimi zvolenou operaci a vypíše výstup. Je ošetřeno i dělení nulou. Program může provádět opakované výpočty, pokud uživatel zadá znak „a“ pro opakování, jinak se program ukončí.

```
{
?znak? <- "a"
DONEC (?znak? = "a") {
  ?vysl? <- 0
  SCRIBO("Zadej prvni cislo: ")
  LECTO(?hod1?)
  SCRIBO("Zadej operaci slovne (plus, minus, krat, deleno): ")
  LECTO(?oper?)
  SCRIBO("Zadej druhe cislo: ")
  LECTO(?hod2?)

  SI (?oper? = "plus") ERGO { ?vysl? <- ?hod1? + ?hod2? } IS
  SI (?oper? = "minus") ERGO { ?vysl? <- ?hod1? - ?hod2? } IS
  SI (?oper? = "krat") ERGO { ?vysl? <- ?hod1? * ?hod2? } IS
  SI (?oper? = "deleno") ERGO {
    SI (?hod2? <> 0) ERGO {
      ?vysl? <- ?hod1? / ?hod2? } CETERA {
        SCRIBO("Nelze delit nulou!")
      } IS
    } IS
  } IS

  SCRIBO("Vysledek je ")
  SCRIBO(?vysl?)
  SCRIBO("
Dalsi vypocet? (a = ano, jinak konec): ")
  LECTO(?znak?)
}
SCRIBO("AHOJ!")
}
```

7.2 Pyramida

Program si vyžádá od uživatele číslo, které reprezentuje počet znaků v základně pyramidy a dále znak, z něhož bude pyramida sestavena. Poté vypíše pyramidu tak, že na vrcholu je jeden znak, ve druhém patře dva atd. Pyramida je symetrická a zarovnaná k levému okraji.

```
{
SCRIBO("Zadej pocet znaku zakladny (male kladne cislo): ")
LECTO(?pocet?)
SCRIBO("Zadej znak: ")
LECTO(?znak?)
SCRIBO ("
")

?patro? <- 1
DONEC (?patro? <= ?pocet?){ [zacatek tvorby pyramidy]
  ?n? <- ?pocet? - ?patro?
  ?poczn? <- ?patro? -1

  DONEC (?n? > 0) { [napise mezery]
    SCRIBO(" ")
    ?n? <- ?n?-1
  } [konec mezer]

  SCRIBO(?znak?) [jeden znak]
  SI (?patro? > 1) ERGO {
    DONEC (?poczn? > 0) {
      SCRIBO(" ")
      SCRIBO(?znak?)
      ?poczn? <- ?poczn?-1
    } [dalsi znaky]
  } IS
  SCRIBO ("
")
  ?patro? <- ?patro? +1
} [konec tvorby pyramidy]
SCRIBO ("AHOJ!")
}
```

7.3 Násobilka

Program vypisuje malou násobilku v deseti tabulkách jdoucích po sobě tak, že nejprve se vypisují násobky čísla 1, poté čísla 2 až po číslo 10.

```
{
  ?cis2? <- "1,000000"
  DONEC (?cis2? <= 10) { [DONEC 1]
    ?cis1? <- "1,000000"
    DONEC (?cis1? <= 10) {[DONEC 2]
      SCRIBO(?cis1?)
      SCRIBO(" * ")
      SCRIBO(?cis2?)
      SCRIBO(" = ")
      SCRIBO(?cis1?*cis2?)
      SCRIBO("
    ")
    ?cis1? <- ?cis1? + 1
  }[ konec DONEC 2]

  ?cis2? <- ?cis2? + 1
  SCRIBO("
")
}[ konec DONEC 1]

SCRIBO("AHOJ!")
}
```

7.4 Faktoriál cyklem

Program obsahuje funkci pro výpočet faktoriálu cyklem. Program vyzve uživatele k zadání čísla, z něž chce počítat faktoriál. Pokud je číslo větší než nebo rovno nule, začne se provádět funkce. Jestliže je číslo 0 nebo 1, výsledkem je jednička (vrací se v proměnné ?REVERSI0?). Při vyšších číslech se počítá pomocí cyklu a výsledná hodnota se opět předá pomocí proměnné ?REVERSI0?. Na závěr se výsledek vypíše. (Vnitřně jsou číselné výpočty prováděny s přesností float.)

```
{
FUNCTIO Fakt (?count?){
  ?ff? <- 1
  ?REVERSI0? <- 1
```



```

DONEC (?count? > 1){
  ?ff? <- ?ff? * ?count?
  ?count? <- ?count? -1
  ?REVERSIO? <- ?ff?
}
} [konec definice fce Fakt]

SCRIBO("Zadej kladne cislo pro vypocet faktorialu: ")
LECTO(?n?)
SI (?n? >= 0) ERGO {
  ?f? <- Fakt(?n?)
  SCRIBO ("Faktorial cisla ")
  SCRIBO (?n?)
  SCRIBO (" je ")
  SCRIBO (?f?)
} CETERA { SCRIBO ("Znovu a neco lepsiho!!!")
} IS

SCRIBO("AHOJ!")
}

```

7.5 Faktoriál rekurzí

Program obsahuje funkci pro výpočet faktoriálu rekurzí. Program vyzve uživatele k zadání čísla, z něž chce počítat faktoriál. Pokud je číslo větší než nebo rovno nule, program uživatele pochválí. Jestliže je možno počítat faktoriál, zavolá se funkce Fakt se zadaným parametrem, která známým rekurzivním způsobem vyčíslí faktoriál a vrátí jeho hodnotu pomocí proměnné ?REVERSIO?. Na závěr se výsledek vypíše.

```

{
  FUNCTIO Fakt (?x?){
    SI (?x? < 2) ERGO {?REVERSIO? <- 1} CETERA {
      ?REVERSIO? <- ?x? * Fakt(?x?-1)} IS
  } # konec funkce Fakt

  SCRIBO("Zadej kladne cislo pro vypocet faktorialu: ")
  LECTO(?n?)
  SI (?n? >= 0) ERGO { SCRIBO("OK")
    } CETERA { SCRIBO("Znovu a neco lepsiho!")
  }
}

```

```

} IS

?f? <- Fakt(?n?)
SCRIBO("Faktorial cisla ")
SCRIBO(?n?)
SCRIBO(" je ")
SCRIBO(?f?)
SCRIBO("
")

SCRIBO("AHOJ!")
}

```

7.6 Hanojské věže

Program řeší známý problém přesunu věže složené z disků mezi dvěma trny. K tomu se používá rekurzivní funkce Prenesvez s parametry, které udávají, odkud kam se který disk přesunuje. Jádrem je informace o tom, odkud kam se disk přesunuje (vzhledem k absenci grafických prostředků je výpis textový). Výstupem je tedy seznam tahů. Nakonec program ještě spočítá a vypíše, kolik tahů bylo.

```

{
  FUNCTIO Prenesvez(?vyska?;?odkud?;?kam?;?pomoci?){
    SCRIBO(?vyska?)
    SI (?vyska? > 0) ERGO{
      Prenesvez(?vyska?-1;?odkud?;?pomoci?;?kam?)
      SCRIBO(?odkud?)
      SCRIBO(" -> ")
      SCRIBO(?kam?)
      SCRIBO("
")
    }
    Prenesvez(?vyska?-1;?pomoci?;?kam?;?odkud?)
  } IS
} # konec funkce Prenesvez

SCRIBO("Zadej pocet disku (male kladne cislo): ")
LECTO(?pocet?)
Prenesvez(?pocet?;1;2;3)

# vypocet poctu tahu (umocneni cyklem)

```

```

?poc? <- 1
SCRIBO (?poc?)
?tah? <- 2
DONEC (?poc? < ?pocet?){
    ?tah? <- ?tah? * 2
    ?poc? <- ?poc? + 1
}
?tah? <- (?tah? - 1)
# tahu je 2 na n-tou - 1
SCRIBO("
Pocet tahu je ")
SCRIBO(?tah?)

SCRIBO("
")
SCRIBO("AHOJ!")
}

```